SVD and Higher Order Correlations for Distributed Data

Ravi Kannan, Santosh Vempala, David Woodruff

June 15, 2014

Distributed Matrices

- **Input Matrix A distributed among s servers.**

Distributed Matrices

- **Input Matrix A distributed among s servers.**
- **Example: Customer-Product matrix**

Distributed Matrices

- **Input Matrix A distributed among s servers.**
- **Example: Customer-Product matrix**
  - **For $t = 1, 2, \ldots, s$, Bundle of goods bought by customers from the t th shop is stored in server t. Call server t 's matrix A^t.**

Distributed Matrices

- **Input Matrix A distributed among s servers.**
- **Example: Customer-Product matrix**
  - **For $t = 1, 2, \ldots, s$, Bundle of goods bought by customers from the t th shop is stored in server t. Call server t 's matrix A^t.**
  - **Customer-Product Matrix $A = A^1 + A^2 + \cdots + A^t$.**

Distributed Matrices

- **Input Matrix A distributed among s servers.**
- **Example: Customer-Product matrix**
  - **For $t = 1, 2, \ldots, s$, Bundle of goods bought by customers from the t th shop is stored in server t. Call server t 's matrix A^t.**
  - **Customer-Product Matrix $A = A^1 + A^2 + \cdots + A^t$.**
  - **More general than the row-partition model in which each customer shops in only one shop.**

The Low Rank Approximation (LRA) Problem

- **Input: $n \times d$ matrix A stored on s servers.**

The Low Rank Approximation (LRA) Problem

- **Input: $n \times d$ matrix A stored on s servers.**
  - **Server t has an $n \times d$ matrix A^t.**

The Low Rank Approximation (LRA) Problem

- **Input: $n \times d$ matrix $\mathbf{A}$ stored on s servers.**
  - **Server t has an $n \times d$ matrix $\mathbf{A}^t$.**
  - **$\mathbf{A} = \mathbf{A}^1 + \mathbf{A}^2 + \cdots + \mathbf{A}^s$.**

The Low Rank Approximation (LRA) Problem

- **Input: $n \times d$ matrix A stored on s servers.**
  - **Server t has an $n \times d$ matrix A^t.**
  - $A = A^1 + A^2 + \cdots + A^s$.
  - **We assume $n \geq d$. Tall Skinny matrix.**

The Low Rank Approximation (LRA) Problem

- **Input: $\mathbf{n} \times \mathbf{d}$ matrix $\mathbf{A}$ stored on $\mathbf{s}$ servers.**
  - **Server $\mathbf{t}$ has an $\mathbf{n} \times \mathbf{d}$ matrix $\mathbf{A}^{\mathbf{t}}$.**
  - **$\mathbf{A} = \mathbf{A}^1 + \mathbf{A}^2 + \cdots + \mathbf{A}^{\mathbf{s}}$.**
  - **We assume $\mathbf{n} \geq \mathbf{d}$. Tall Skinny matrix.**
- **Output: Server $\mathbf{t}$ has $\mathbf{n} \times \mathbf{d}$ matrix $\mathbf{C}^{\mathbf{t}}$ satisfying**

The Low Rank Approximation (LRA) Problem

- **Input: $n \times d$ matrix A stored on s servers.**
  - **Server t has an $n \times d$ matrix A^t.**
  - $A = A^1 + A^2 + \cdots + A^s$.
  - **We assume $n \geq d$. Tall Skinny matrix.**
- **Output: Server t has $n \times d$ matrix C^t satisfying**
  - $C = \sum_{t=1}^{s} C^t$ **has rank at most k;**

The Low Rank Approximation (LRA) Problem

- **Input: $n \times d$ matrix A stored on s servers.**
  - **Server t has an $n \times d$ matrix A^t.**
  - **$A = A^1 + A^2 + \cdots + A^s$.**
  - **We assume $n \geq d$. Tall Skinny matrix.**
- **Output: Server t has $n \times d$ matrix C^t satisfying**
  - **$C = \sum_{t=1}^{s} C^t$ has rank at most k;**
  - **$||A - C||_F \leq (1+\varepsilon)||A - C^*||_F$, C^*= Best Possible Rank k approx to A.**

The Low Rank Approximation (LRA) Problem

- **Input: $n \times d$ matrix A stored on s servers.**
  - **Server t has an $n \times d$ matrix A^t.**
  - **$A = A^1 + A^2 + \cdots + A^s$.**
  - **We assume $n \geq d$. Tall Skinny matrix.**
- **Output: Server t has $n \times d$ matrix C^t satisfying**
  - **$C = \sum_{t=1}^{s} C^t$ has rank at most k;**
  - **$||A - C||_F \leq (1 + \varepsilon)||A - C^*||_F$, C^*= Best Possible Rank k approx to A.**
- **Resources: Each Server is Poly time, Linear Space. Communication in $O(1)$ rounds. Bound the total number of words communicated.**

The Low Rank Approximation (LRA) Problem

- **Input: $n \times d$ matrix A stored on s servers.**
  - **Server t has an $n \times d$ matrix A^t.**
  - **$A = A^1 + A^2 + \cdots + A^s$.**
  - **We assume $n \geq d$. Tall Skinny matrix.**
- **Output: Server t has $n \times d$ matrix C^t satisfying**
  - **$C = \sum_{t=1}^{s} C^t$ has rank at most k;**
  - **$||A - C||_F \leq (1 + \varepsilon)||A - C^*||_F$, C^*= Best Possible Rank k approx to A.**
- **Resources: Each Server is Poly time, Linear Space. Communication in $O(1)$ rounds. Bound the total number of words communicated.**
- **Main Theorems of this part: Near-Optimal Communication Algorithm : $\Theta^*(skd)$. No n !**

Random Matrix I

- **Clarkson and Woodruff** $m \in O^*(k)$. **If** S **is a** $m \times n$ **random sign matrix and** U **is the projection onto row space of** SA**, then, the best rank** k **approximation to** AU **does the job.**

Random Matrix I

- **Clarkson and Woodruff** $m \in O^*(k)$. If S is a $m \times n$ random sign matrix and U is the projection onto row space of SA, then, the best rank k approximation to AU does the job.
- **Proposed Algorithm:**
  - Central Processor (CP) selects S and sends it to each server.

Random Matrix I

- Clarkson and Woodruff $m \in O^*(k)$. If S is a $m \times n$ random sign matrix and U is the projection onto row space of SA, then, the best rank k approximation to AU does the job.
- Proposed Algorithm:
  - Central Processor (CP) selects S and sends it to each server.
  - Server t finds SA^t, sends it to CP.

Random Matrix I

- **Clarkson and Woodruff** $m \in O^*(k)$. If S is a $m \times n$ random sign matrix and U is the projection onto row space of SA, then, the best rank k approximation to AU does the job.
- Proposed Algorithm:
  - Central Processor (CP) selects S and sends it to each server.
  - Server t finds SA^t, sends it to CP.
  - CP finds $SA = \sum_t SA^t$, finds U and sends it all servers.

Random Matrix I

- Clarkson and Woodruff $m \in O^*(k)$. If S is a $m \times n$ random sign matrix and U is the projection onto row space of SA, then, the best rank k approximation to AU does the job.
- Proposed Algorithm:
  - Central Processor (CP) selects S and sends it to each server.
  - Server t finds SA^t, sends it to CP.
  - CP finds $SA = \sum_t SA^t$, finds U and sends it all servers.
  - Server t finds A^tU, sends it to CP. CP finds AU, does SVD. Done.

Random Matrix I

- Clarkson and Woodruff $m \in O^*(k)$. If S is a $m \times n$ random sign matrix and U is the projection onto row space of SA, then, the best rank k approximation to AU does the job.
- Proposed Algorithm:
  - Central Processor (CP) selects S and sends it to each server.
  - Server t finds SA^t, sends it to CP.
  - CP finds $SA = \sum_t SA^t$, finds U and sends it all servers.
  - Server t finds A^tU, sends it to CP. CP finds AU, does SVD. Done.
- Sending S : $k-$ wise independence enough. Send only seed of pseudo-random S. Length $O^*(k)$. Sending $A^tU = B^t$ which is $n \times O^*(k)$.

Problem Reduced to

- **Server t has $\mathbf{n} \times \mathbf{k}$ matrix $\mathbf{B^t}$. Do SVD of $\mathbf{B} = \sum_\mathbf{t} \mathbf{B^t}$.**

Problem Reduced to

- **Server t has $n \times k$ matrix B^t. Do SVD of $B = \sum_t B^t$.**
- **(Simple) Lemma: If P is a pseudo-random $O^*(k) \times n$ matrix, for all $x \in R^k$, $|PBx| \approx_\varepsilon |Bx|$, so SVD of PB suffices to solve low-rank approx (LRA) of B.**

Problem Reduced to

- Server t has $n \times k$ matrix B^t. Do SVD of $B = \sum_t B^t$.
- (Simple) Lemma: If P is a pseudo-random $O^*(k) \times n$ matrix, for all $x \in R^k$, $|PBx| \approx_\varepsilon |Bx|$, so SVD of PB suffices to solve low-rank approx (LRA) of B.
- "Suffices" does not immediately give LRA to A. See paper for full proof.

Problem Reduced to

- Server t has $n \times k$ matrix B^t. Do SVD of $B = \sum_t B^t$.
- (Simple) Lemma: If P is a pseudo-random $O^*(k) \times n$ matrix, for all $x \in R^k$, $|PBx| \approx_\varepsilon |Bx|$, so SVD of PB suffices to solve low-rank approx (LRA) of B.
- "Suffices" does not immediately give LRA to A. See paper for full proof.
- This two-stage adaptive sketching process has since played an important role in several follow-up works like - CUR Matrix Factorization by Boutsides and Woodruff.

Problem Reduced to

- Server t has $n \times k$ matrix B^t. Do SVD of $B = \sum_t B^t$.
- (Simple) Lemma: If P is a pseudo-random $O^*(k) \times n$ matrix, for all $x \in R^k$, $|PBx| \approx_\varepsilon |Bx|$, so SVD of PB suffices to solve low-rank approx (LRA) of B.
- "Suffices" does not immediately give LRA to A. See paper for full proof.
- This two-stage adaptive sketching process has since played an important role in several follow-up works like - CUR Matrix Factorization by Boutsides and Woodruff.
- Lower Bound: If server t has now $k \times d$ matrix A^t and finally, we have to have LRA to $A = \sum_t A^t$ on one server, clearly need $\Omega(skd)$ communication. But here, we only need server t to end up with C^t, so sum of C^t is an LRA.

Problem Reduced to

- Server t has $n \times k$ matrix B^t. Do SVD of $B = \sum_t B^t$.
- (Simple) Lemma: If P is a pseudo-random $O^*(k) \times n$ matrix, for all $x \in R^k$, $|PBx| \approx_\varepsilon |Bx|$, so SVD of PB suffices to solve low-rank approx (LRA) of B.
- "Suffices" does not immediately give LRA to A. See paper for full proof.
- This two-stage adaptive sketching process has since played an important role in several follow-up works like - CUR Matrix Factorization by Boutsides and Woodruff.
- Lower Bound: If server t has now $k \times d$ matrix A^t and finally, we have to have LRA to $A = \sum_t A^t$ on one server, clearly need $\Omega(skd)$ communication. But here, we only need server t to end up with C^t, so sum of C^t is an LRA.
- Trick: Server 1 and 2 special- have $A^1 = I_d$ and $A^2 = -I_d$, so $A = \sum_{t \geq 3} A^t$. At end, server 1 has to learn the projection to the LRA subspace which gives away the actual sum.

Other Such Problems not in the paper

- **Interest in Distributed Optimization in ML.**

Other Such Problems not in the paper

- **Interest in Distributed Optimization in ML.**
- **Provably: For Linear Programming :$\mathbf{Ax} \leq \mathbf{b}$, $\mathbf{A_{n \times d}}$, with each constraint residing in one server, $\mathbf{O}^*(\mathbf{d^4})$ communication suffices, but poly(d) rounds. (Ellipsoid Algorithm).**

Other Such Problems not in the paper

- **Interest in Distributed Optimization in ML.**
- **Provably: For Linear Programming :$\mathbf{Ax} \leq \mathbf{b}$, $\mathbf{A_{n \times d}}$, with each constraint residing in one server, $\mathbf{O^*(d^4)}$ communication suffices, but poly(d) rounds. (Ellipsoid Algorithm).**

Other Such Problems not in the paper

- **Interest in Distributed Optimization in ML.**
- **Provably: For Linear Programming :$\mathbf{Ax} \leq \mathbf{b}$, $\mathbf{A_{n\times d}}$, with each constraint residing in one server, $\mathbf{O^*(d^4)}$ communication suffices, but poly(d) rounds. (Ellipsoid Algorithm).**
- **Lower bound for Linear Equations: $\mathbf{Ax} = \mathbf{b}$: $\mathbf{\Omega(d^2)}$ communication.**

Other Such Problems not in the paper

- **Interest in Distributed Optimization in ML.**
- **Provably: For Linear Programming :$\mathbf{Ax} \leq \mathbf{b}$, $\mathbf{A_{n\times d}}$, with each constraint residing in one server, $\mathbf{O^*(d^4)}$ communication suffices, but poly(d) rounds. (Ellipsoid Algorithm).**
- **Lower bound for Linear Equations: $\mathbf{Ax} = \mathbf{b}$: $\mathbf{\Omega(d^2)}$ communication.**
- **Linear and non-linear optimization, fault tolerant SVD, all of interest.**

Higher Order Correlations-Examples

- **Example: Time-series data for many events. Each event resides fully on one server.**

Higher Order Correlations-Examples

- **Example: Time-series data for many events. Each event resides fully on one server.**
- **Estimate to relative error ε number of 7-tuples $(\mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}, \mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}, \mathbf{t_4})$ such that each of $\mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}$ occurs at each of the times $\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}, \mathbf{t_4}$.**

Higher Order Correlations-Examples

- **Example: Time-series data for many events. Each event resides fully on one server.**
- **Estimate to relative error ε number of 7-tuples $(\mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}, \mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}, \mathbf{t_4})$ such that each of $\mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}$ occurs at each of the times $\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}, \mathbf{t_4}$.**
- **Bipartite graph. Estimate number of $K_{3,4}$.**

Higher Order Correlations-Examples

- **Example: Time-series data for many events. Each event resides fully on one server.**
- **Estimate to relative error ε number of 7-tuples $(\mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}, \mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}, \mathbf{t_4})$ such that each of $\mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}$ occurs at each of the times $\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}, \mathbf{t_4}$.**
- **Bipartite graph. Estimate number of $K_{3,4}$.**
- **More generally: Each customer resides wholly on one server. Estimate number of $(\mathbf{C_1}, \mathbf{C_2}, \mathbf{C_3}, \mathbf{P_1}, \mathbf{P_2}, \mathbf{P_3}, \mathbf{P_4})$ such that each of the 3 customers C_1, C_2, C_3 buys at least x amount of at least 3 of the 4 products $\mathbf{P_1}, \mathbf{P_2}, \mathbf{P_3}, \mathbf{P_4}$.**

Higher Order Correlations-Examples

- **Example: Time-series data for many events. Each event resides fully on one server.**
- **Estimate to relative error ε number of 7-tuples $(\mathbf{E_1, E_2, E_3, t_1, t_2, t_3, t_4})$ such that each of $\mathbf{E_1, E_2, E_3}$ occurs at each of the times $\mathbf{t_1, t_2, t_3, t_4}$.**
- **Bipartite graph. Estimate number of $K_{3,4}$.**
- **More generally: Each customer resides wholly on one server. Estimate number of $(\mathbf{C_1, C_2, C_3, P_1, P_2, P_3, P_4})$ such that each of the 3 customers C_1, C_2, C_3 buys at least x amount of at least 3 of the 4 products $\mathbf{P_1, P_2, P_3, P_4}$.**
- **Customer i turns into a $\binom{\mathbf{n}}{\mathbf{4}}$ component vector $\mathbf{v_i}$ and answer is (essentially) $||\sum_{\mathbf{i}} \mathbf{v_i}||_{\mathbf{3}}^{\mathbf{3}} =$ Sum of cubes of components of the sum of vectors $\mathbf{v_i}$.**

Higher Order Correlations-Examples

- **Example: Time-series data for many events. Each event resides fully on one server.**
- **Estimate to relative error ε number of 7-tuples $(\mathbf{E_1, E_2, E_3, t_1, t_2, t_3, t_4})$ such that each of $\mathbf{E_1, E_2, E_3}$ occurs at each of the times $\mathbf{t_1, t_2, t_3, t_4}$.**
- **Bipartite graph. Estimate number of $K_{3,4}$.**
- **More generally: Each customer resides wholly on one server. Estimate number of $(\mathbf{C_1, C_2, C_3, P_1, P_2, P_3, P_4})$ such that each of the 3 customers C_1, C_2, C_3 buys at least x amount of at least 3 of the 4 products $\mathbf{P_1, P_2, P_3, P_4}$.**
- **Customer i turns into a $\binom{\mathbf{n}}{\mathbf{4}}$ component vector $\mathbf{v_i}$ and answer is (essentially) $||\sum_{\mathbf{i}} \mathbf{v_i}||_{\mathbf{3}}^{\mathbf{3}} =$ Sum of cubes of components of the sum of vectors $\mathbf{v_i}$.**
- **In streaming, lower bounds tell us that the 3 above (> 2) makes it impossible with polylog communication. But in this distributed model, the lower bound does not apply and we can in fact "do it".**

Higher Order Correlations

- **General Set-up: A set of n$-$ vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots$, each wholly residing in one server.**

Higher Order Correlations

- **General Set-up: A set of $\mathbf{n}-$ vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots$, each wholly residing in one server.**
- **$\mathbf{k}, \mathbf{r}$ fixed positive integers. $\mathbf{g} : \mathbf{R}_+^{\mathbf{k}} \to \mathbf{R}_+$, any monotone function.**

Higher Order Correlations

- **General Set-up: A set of $n-$ vectors $v_1, v_2, \ldots$, each wholly residing in one server.**
- **k, r fixed positive integers. $g : R_+^k \to R_+$, any monotone function.**
- **$\sum_{j_1, j_2, \ldots, j_k \in [n]} \left(\sum_i g\left(v_{i,j_1}, v_{i,j_2}, \ldots, v_{i,j_k}\right)\right)^r$ can be estimated to relative error $\varepsilon > 0$ in linear space, poly time, $O(1)$ rounds and total communication $O^*(s^{r+2})$.**

Higher Order Correlations

- **General Set-up: A set of $n-$ vectors $v_1, v_2, \ldots$, each wholly residing in one server.**
- **k, r fixed positive integers. $g : R_+^k \to R_+$, any monotone function.**
- **$\sum_{j_1, j_2, \ldots, j_k \in [n]} \left(\sum_i g\left(v_{i,j_1}, v_{i,j_2}, \ldots, v_{i,j_k}\right)\right)^r$ can be estimated to relative error $\varepsilon > 0$ in linear space, poly time, $O(1)$ rounds and total communication $O^*(s^{r+2})$.**
- **Further, we need $\Omega(s^{r-1})$ and the gap of s^3 can be closed when $k = 1$. [Classic problem for streaming-"frequency moments"]**

Idea of Algorithm, Generalizations

- **Server t has vector $\mathbf{u^t}$. Want $||\sum_t \mathbf{u^t}||_3^3$.**

Idea of Algorithm, Generalizations

- **Server t has vector $\mathbf{u}^t$. Want $||\sum_t \mathbf{u}^t||_3^3$.**
- **Enough to sample i with probabilities proportional to $(\sum_t \mathbf{u}_i^t)^3$.**

Idea of Algorithm, Generalizations

- **Server t has vector $\mathbf{u}^t$. Want $||\sum_t \mathbf{u}^t||_3^3$.**
- **Enough to sample i with probabilities proportional to $(\sum_t \mathbf{u}_i^t)^3$.**
- **Basic Idea: Server t samples i with probabilities proportional to $(\mathbf{u}_i^t)^3$.**

Idea of Algorithm, Generalizations

- **Server t has vector u^t. Want $||\sum_t u^t||_3^3$.**
- **Enough to sample i with probabilities proportional to $(\sum_t u_i^t)^3$.**
- **Basic Idea: Server t samples i with probabilities proportional to $(u_i^t)^3$.**
- **Everyone sends sample to CP. Some careful rejection sampling.**

Idea of Algorithm, Generalizations

- **Server t has vector u^t. Want $||\sum_t u^t||_3^3$.**
- **Enough to sample i with probabilities proportional to $(\sum_t u_i^t)^3$.**
- **Basic Idea: Server t samples i with probabilities proportional to $(u_i^t)^3$.**
- **Everyone sends sample to CP. Some careful rejection sampling.**
- **In final theorem, more general functions than r th power are dealt with.**

Idea of Algorithm, Generalizations

- Server t has vector u^t. Want $||\sum_t u^t||_3^3$.
- Enough to sample i with probabilities proportional to $(\sum_t u_i^t)^3$.
- Basic Idea: Server t samples i with probabilities proportional to $(u_i^t)^3$.
- Everyone sends sample to CP. Some careful rejection sampling.
- In final theorem, more general functions than r th power are dealt with.
- Open: Charecterize all functions f, g which can be handled.